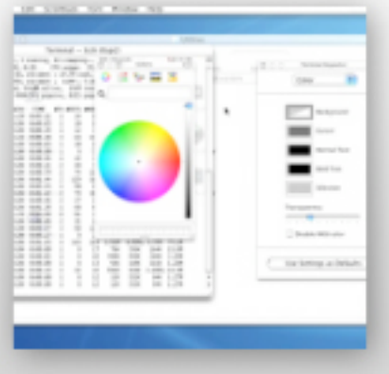
Mac OS X Security

Architecture Overview

Session 101



Mac OS X Security
Architecture Overview

Craig Keithley
Security Technology Evangelist
keithley@apple.com

Mac OS X Security
Architecture Overview

Perry The Cynic
Apple Security Group
perry@apple.com

Introduction: Weird Stuff

- Rarely the main feature
- Sometimes a valuable addition
- Usually just gets in the way
- **Weakest Link Effect**

What You'll Learn

- What security facilities are there in Mac OS X?
- How are they structured?
- What to use for what problem?
- How to use them?
- Pitfalls and caveats

Technologies: Standard

- Standard UNIX facilities
- Standard Mach 3 facilities
- CDSA cryptographic API
- Kerberos 5
- Many Open Source libraries (ssh, openssl, etc.)

Technologies: Mac OS X Specific

- Keychain storage
- Authorization facility
- Certificate and Trust Management
- Directory Services

Standard UNIX Security Facilities

- Separation by address spaces (processes)
- User and group ids
  - File system permissions
  - Root user id
- Setuid/setgid facility

About UNIX Root Privilege

- Root privilege is potentially very dangerous
- Prefer specific facilities
- Use an expert or **be** an expert
  - KISS—Simple and Small
  - **Factor** your program

What's Unusual for a UNIX System?

- There's Mach in there!
  - Un-UNIX features alongside UNIX
  - Unexpected ways to get UNIX results
- Root user disabled (by default)
- Admin users are "almost root"

Standard Mach 3 Facilities

- Not recommended to most developers
- Mach IPC facility
- Mach ports = access rights
- Bootstrap port facility used extensively

CDSA Standard Cryptography

- Open Group standard
- Full implementation in Mac OS X
  - Written from scratch
  - Open Sourced
- All official APIs use CDSA
- Accessible from higher-layer APIs

Do You Want to Call CDSA Directly?

- Good news
  - Very powerful
  - Extensible, flexible, configurable
- Bad news
  - Steep learning curve (conventions, rules)
  - Lots of (C) code, verbose
- "As needed"
  - Use higher layers, drop down as needed

Keychains: Overview

- Safe storage of secrets (passwords, keys, etc.)
- Encrypted storage containers
- Protected with passphrase (or other secret)
- Controlled item access
- Offline-safe

Keychains: Architecture

- Database with extensible schemas
- Managed by system daemon (SecurityServer)
- “Hands-off” handling of secrets
- Scalable APIs

Keychains: Security Profile

- UNIX access permissions (as usual)
- Master unlock (passphrase et al)
- Access control lists for protected items
- Need not trust file system

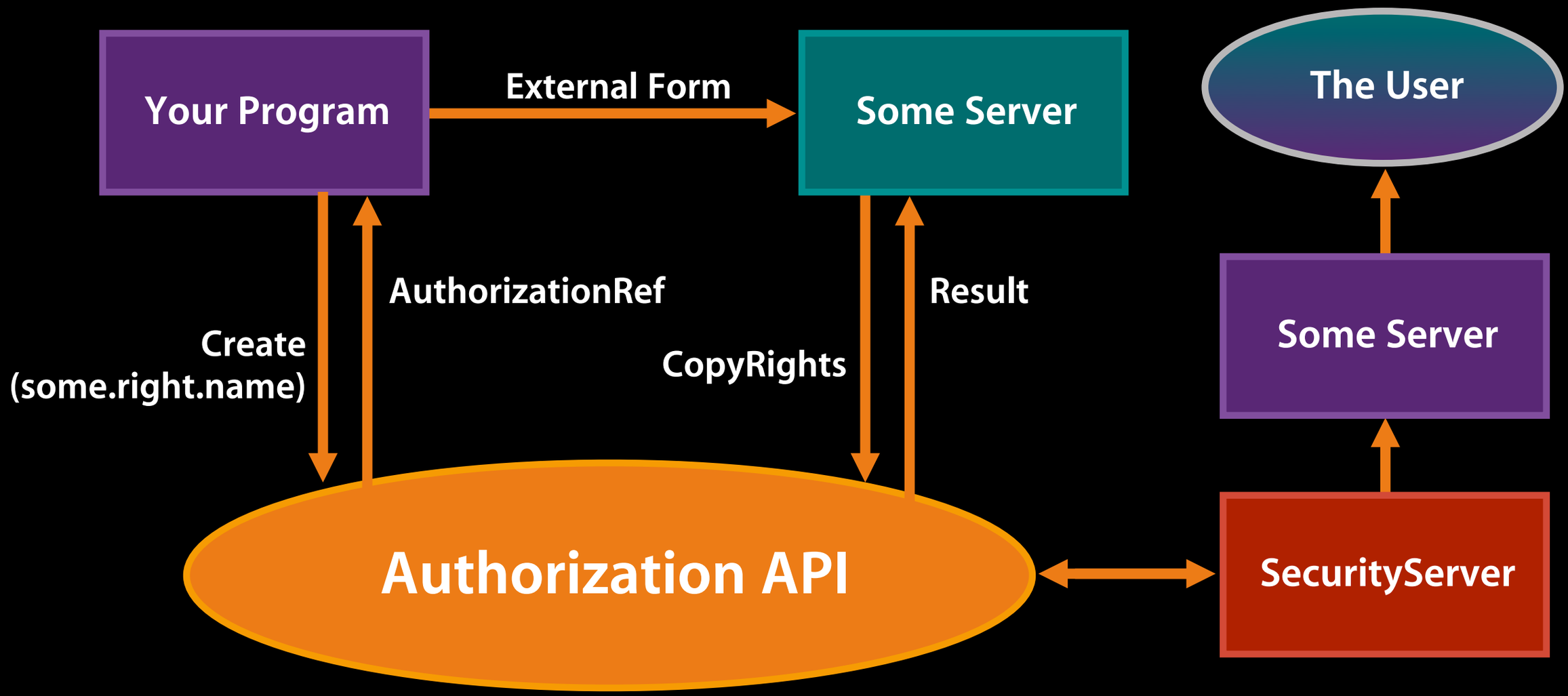
Keychains: Advanced

- Any number, anywhere, shared
- Operate on keychain search lists
- Portable
- Extensible item types (schemata)
- System Keychains

Authorization API

- Authorization, not capability
- Per-system configuration by administrator
- Supports selective **Single Sign-On**
- Extensible with plugins
- Two-way interfaced with PAM

Authorization: Overview



Authorization Components

- Rights
  - Abstract rights named by hierarchical strings
    - "system.privilege.admin" (become root)
    - "com.frobozz.firefax.multicast" (perhaps one of yours)
- Credentials
  - Things you have or are (and can prove)
  - Shareable and persistent
  - E.g., administrator access check

Authorization Components (Cont.)

- Rules
  - System wide, set by administrator
  - Maps rights to required credentials
  - Configuration API

Authorization-Based Facilities

- Preference panes (“lock icon”)
- Most “admin checks”
- AuthorizationExecuteWithPrivileges
- AuthorizationBindPrivilegedPort

Certificate and Trust Management

- X509 certificate standard
  - Store certificates in keychain files
  - System default anchor set
  - High-level SecTrust API for evaluation
  - UserTrust database for persistent trust choices

SecureTransport (SSL)

- Full Implementation of SSL standard
- Based on CDSA cryptography
- Ties into SecTrust/UserTrust
- Not lenient by default(!)
- Accessible through CFNetwork (https: schema)
- Apple's official SSL solution—use it!

CMS + S/MIME (preliminary)

- Cryptographic Message System API
- S/MIME support
- Based on SecIdentity and SecTrust
- **Preliminary** support in Panther

Encrypted Disk Images

- Disk images can be optionally encrypted
- CDSA for cryptography
- Uses passphrases
  - Acquired out of band
  - Stored in keychain
  - Re-fetched on system wake-up

Strategies and Approaches

- Find the right API(s)
- Higher (simpler) is better
- Keep it simple, stay on the marked path
- Think early, build it in

What Feature for Which Problem?

Keep Secrets	Key Chains
Provide privileged services	Authorization
Network transport security	SecureTransport
Network authentication	Directory Services
Certificates	CMS
Cryptography	CDSA
Real life: Mix and Match	

Strategies: DO

- Use recommended Mac OS X facilities
  - Authorization
  - SecureTransport
  - SecTrust
- Store secrets in keychains
- Use available facilities for privileged operations
- Factor your application (with Authorization!)

Strategies: DON’T

- Store passwords and secrets in plain files
- Invent your own authentication schemes
- From privileged (factored) code
  - Load plugins
  - Link against GUI libraries
  - Call dangerous functions (system, popen)
- Assume there is (only) one logged-in user
- Assume keychains are always files

Future-Proofing Your Code

- Use Mac OS X standard APIs
  - Don’t bypass or mangle them
  - Don’t copy code blindly
- Don’t assume access is password based
- Don’t assume there is (only) one user on
- Test with different security configurations
  - Authorization configuration
  - Keychain item ACLs

Security in Darwin vs. Mac OS X

- Almost all **Darwin**
  - CDSA with all plugins
  - SecurityServer
  - Authorization engine
- Graphic facilities are Mac OS X only
  - User interaction and prompting
  - Stubbed out in Darwin

What's New for Panther?

- Certificate Support
  - CMS and S/MIME (preliminary)
  - PKCS7/12 import/export
  - SecureTransport uses SecTrust
- System Keychains
- New Cocoa APIs (SecurityFoundation, SecurityInterface)

What's New for Panther? (Cont.)

- Application change tracking (fewer dialogs)
- Authorization
  - Access to low-numbered ports
  - Custom-add API
  - More flexible dialogs
- Major improvements in Keychain Access
- Secure erase

Wrap Up

Roadmap

102 Open Source at Apple Sample description	**The Mission** **Tue, 10:30pm**
108 Kerberos in Mac OS X Sample description	**The Mission** **Thu, 9:00am**
109 Security: Certificate APIs Sample description	**The Marina** **Thu, 10:30am**
FF001 Darwin Feedback Forum Sample description	**North Beach** **Tue, 3:30pm**
FF016 Security Feedback Forum Sample description	**North Beach** **Fri, 2:00pm**

Who to Contact

Craig Keithley
Security Technology Evangelist
keithley@apple.com

Perry The Cynic
Apple Security Group
perry@apple.com

http://developer.apple.com/wwdc2003/urls.html

For More Information

Reference Library

Documentation

Security Overview for Software Developers

Reference Library

Documentation

Mac OS X Security (general)

http://developer.apple.com/techpubs/macosx/CoreTechnologies/coretechnologies.html

Keychain API

http://developer.apple.com/techpubs/macosx/CoreTechnologies/securityservices/keychainservices/keychainservices.html

Authorization API

http://developer.apple.com/techpubs/macosx/CoreTechnologies/securityservices/authorizationservices/authservices.html

Reference Library

Other Resources

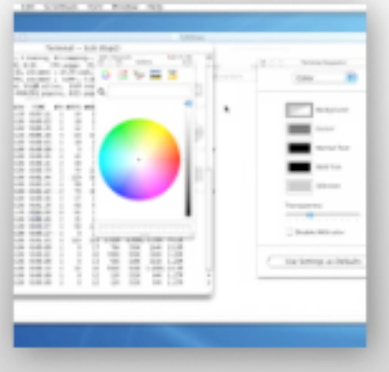
ADC Security page
http://developer.apple.com/security/index.html

Apple CDSA mailing list
http://lists.apple.com/mailman/listinfo/apple-cdsa

Common Data Security Architecture (CDSA)
http://www.opensource.apple.com
http://www.opengroup.org

Secure Trusted OS Consortium
http://www.stosdarwin.org

Q&A



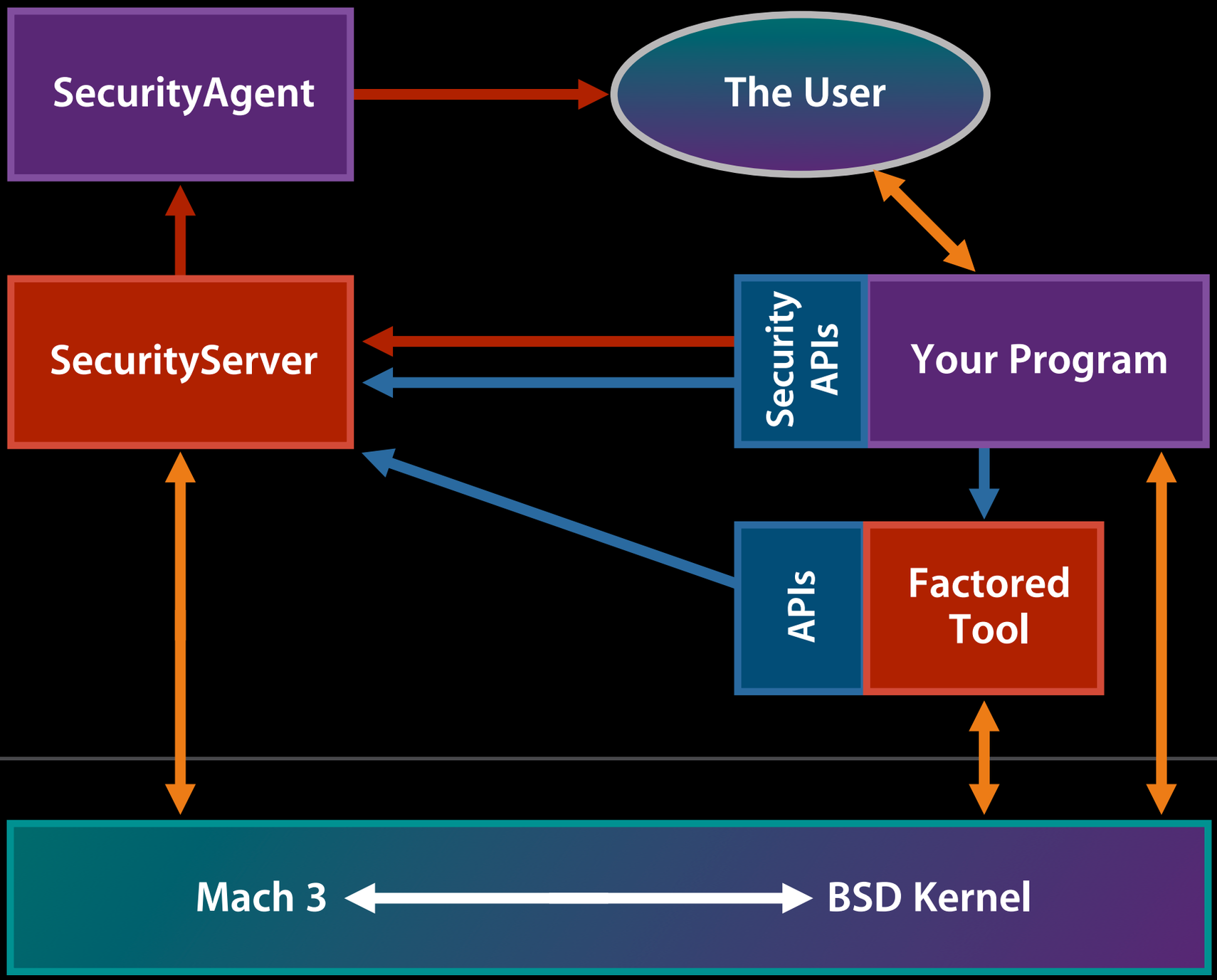
Craig Keithley
Security Technology Evangelist
keithley@apple.com

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference

A Note on Openssl

- Openssl library is included in Mac OS X
- Fine for straight ports
- Don’t use it for Mac OS X specific solutions
  - Raw keys
  - A tool library, not an architecture
  - Has no hooks to higher-layer facilities (e.g., Trust)

The Big Picture
(Don't Panic)